基于 SEDA 企业服务总线负载控制①

马　存[1,2], 马　跃[2], 廉东本[2], 李文博[2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院 沈阳计算技术研究所, 辽宁 沈阳 110171)

摘　要: 在基于分阶段事件驱动架构下, 把企业服务总线系统按事务逻辑处理功能划分成消息监听、协议解析、消息转换、消息路由、消息发送五个阶段. 系统在运行过程中, 由中央调度器对各个阶段的负载信息进行收集. 本文根据均值方差分析, 对各个阶段的线程池大小进行动态调整, 使得每个阶段的处理事务能力协调关联起来, 从而避免出现由于某个阶段负载过重而影响整个系统的性能.

关键词: SOA; 企业服务总线; 阶段事件驱动架构; 均值方差分析

Load Control of SEDA-Based Enterprise Service Bus

MA Cun[1,2], MA Yue[2], LIAN Dong-Ben[2], LI Wen-Bo[2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology of Chinese Academy of Sciences, Shenyang 110171, China)

Abstract: The Enterprise Service Bus system which is based on Stage Event-Driven Architecture is divided into five stages according to the logic processing functions: message listener, protocol analysis, message exchange, message route and message send. During the system running, the central scheduler collects states of these stages' information. This research adjusts thread pool size of each stage dynamically based on the mean-variance analysis, then all stages' task processing ability are associated, which can avoid some stages' heavily burden that may affects the performance of entire system.

Key words: SOA; enterprise service bus; service event-driven agriculture; mean-variance analysis

企业服务总线(Enterprise Service Bus, ESB)是用来整合应用和服务的一个灵活的基础架构, 是SOA(Service Oriented Architecture)的一种实现方式[1]. ESB 实质上是服务间的连接框架, 其核心功能包括消息监听、协议解析、消息转换、消息路由等部分[2].

目前, 市场上有很多 ESB 产品, 其中 IBM 和 Oracle 在市场上占有领先地位, 但其技术相对封闭. MuleSoft 的开源 ESB 影响力也比较大, 它是采用消息代理机制模式, 消息传送是基于分阶段事件驱动架构(Service Event-Driven Agriculture, SEDA)架构实现的, 其每个阶段都是一个组件, 组件之间通过事件队列进行联系. 但是在服务请求高并发的情况下, 它并没有设计出一个合理的针对全局负载信息调度控制器, 对各个阶段负载情况进行收集并调节各个阶段线程池的大小. ESB 产品在国内方面研究的相对较少, 主要集中在高校和少数企业内, 软件通用性较差.

本文将 SEDA 架构思想应用到企业服务总线框架中, 针对缺少全局调度控制信息的情况下, 设计一个基于反馈调节和负载均衡的企业服务总线系统.

1　SEDA架构和阶段负载控制

SEDA 是加州大学伯克利分校的 Matt Welsh 博士提出一套解决高并发、高吞吐率的服务器程序架构模型[3]. SEDA 的核心思想是将应用系统按照业务逻辑划分为一系列相关阶段, 对于消耗不同资源的每个阶段使用不同数量的线程来处理. 每个阶段都是一个相对

① 基金项目:国家水体污染控制与治理科技重大专项(2012ZX07505)
收稿时间:2013-05-08;收到修改稿时间:2013-06-06

独立的模块, 分别完成一个独立的逻辑功能单元. 阶段之间用事件队列连接起来, 所有阶段组成了一个完整的工作流[4]. 在此基础上, 本文添加了中央调度控制器和阶段调度控制器.

1.1 SEDA 的阶段组成

事件和事件队列: 客户端的一次服务请求在企业服务总线中表示一次任务. 任务在每个阶段的表现形式是一个事件, 而事件队列是事件的集合, 它有队列大小和当前等待事件个数的属性.

事件处理器: 是各阶段的核心模块, 且每个阶段都有自己的事件处理器, 并继承统一的事件处理器接口.

线程池: 它以多线程的机制并发驱动事件处理器的执行. 线程池的大小由本阶段的调度控制器进行设置.

调度控制器: 在系统运行过程中, 调度控制器保存着本阶段的负载情况, 并由中央调度器统一收集, 之后再根据中央调度器的反馈情况设置本阶段的线程池大小和事件队列长度.

中央调度控制器: 在系统运行过程中, 中央调度器定时收集各个阶段的调度控制器的运行负载信息, 经过集中计算, 再把信息反馈给每个阶段.

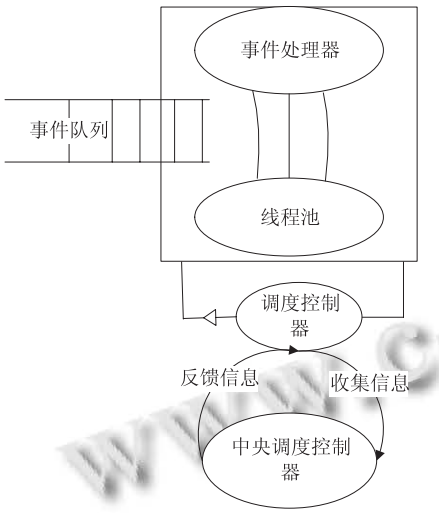

图 1 SEDA 阶段框架图

1.2 阶段负载控制研究

在根据 1.1 节描述的阶段结构中, 当外部事件(相对于本阶段的上一个阶段)进入本阶段事件队列时, 就会通知阶段线程池. 若线程池里有空闲线程, 就把事件队列里的事件取出, 交给事件处理器处理. 根据事件队列和线程池的关系, 中央控制器在收集阶段信息时, 先取本阶段线程池中空闲线程数, 若有空闲线程, 就认为事件队列里没有等待事件; 若没有空闲线程, 就到事件队列里取等待事件个数. 如表 1 所描述:

表 1 阶段资源情况

阶段编号	1	2	……	n
可用线程数	5	-3	……	2

在表 1 中, 正数表示空闲线程个数; 负数表示此阶段没有空闲线程, 且事件队列中仍有相应数目的事件等待处理. 当可用线程数表示当前阶段的负载情况, 用符号 s 表示当前阶段的可用线程个数.

阶段占用系统资源系数用 p 表示, 假设以第一阶段 p_1=1 为基准, 阶段线程池中线程个数用符号 t 表示, 其它阶段线程池大小和第一阶段线程池大小比值就是相应其它阶段的 p 值. 比如第一阶段线程池大小 t_1=10, 第二阶段线程池大小 t_2=15, 那么第二阶段的 p 值应为 $p_2= t_2/t_1$=1.5. 系统在启动时, 线程池的线程个数初始化为统一大小, 系统在运行过程中根据实际情况自行调整. 这种设置方便模块的集成, 开发人员不用过多关心每个模块在整个系统中所占资源的比例. p 值的计算方法如下:

$$p_i = \frac{t_i}{t_1}\left(1 < i <= n\right) \tag{1}$$

各个阶段资源总和(R):

$$R = \sum_1^n s_i * p_i \tag{2}$$

R 作为一个系统资源负载的指数. 如果 R 值较大, 表明系统资源充足; 若为负数, 表明系统处于繁忙状态. 再根据以下公式判断各阶段负载是否均衡.

系统负载均值 u 和方差 v 的计算如下:

$$u = \frac{\sum_1^n s_i * p_i}{n} = \frac{R}{n} \tag{3}$$

$$v = \sum_1^n \left(s_i * p_i - u\right)^2 \Big/ n \tag{4}$$

根据参数 u 和 v 可以得知系统的负载分布情况, 如表 2 所描述:

表 2 系统负载状态表

u \ v	小	大
小	系统忙, 资源分配均匀	服务忙, 资源分配不均
大	系统闲, 资源分配均匀	系统闲, 资源分配不均

阶段资源分配均匀是系统所要追求的设计目标. 因此由上述基于均值方差的分析可见, 各阶段线程池大小需要调整的是负载方差 v 值大的情况.

我们根据系统资源负载指数 R 值和各阶段的占用资源系数 p 来调整各个阶段的线程池大小. 设定调整后的可用资源数均值为 r_0, 如公式 5 所示:

$$r_0 = R\Big/\sum_1^n p_i^2 \tag{5}$$

其中, R 是系统资源总数, 我们把 R 均衡的分配给各个阶段, 分配量由 r_0 表示, 从而可以得出每个阶段的调整后的资源数为:

$$r_i = r_0 * p_i\left(1 < i <= n\right) \tag{6}$$

利用公式 6, 得出各个阶段需要调整的线程个数为 s_i', 计算如下:

$$s_i' = r_i - s_i \tag{7}$$

系统运行过程中, 中央调度控制器把 s_i' 发给各个阶段的调度控制器即可完成调整. 根据阶段流程的事件丢弃损失最小机制原则, 后一个阶段的事件队列溢出要比它之前的阶段事件队列丢弃损失严重. 所以越往后的阶段拥有的系统资源的优先权就越大. 通过上面的分析, 当增加线程时, 先从最后面的阶段开始增加; 在减少线程时, 从最前面的阶段开始减少.

第一阶段是后续阶段事件的提供者, 是入口事件的闸门, 其事件处理能力将会影响到后续的所有阶段. 所以第一阶段的线程池大小将会影响后面阶段的线程池大小. 当有事件到达第一阶段时, 如果事件队列长度超过一定阈值(比如事件队列长度的一半), 那么第一阶段增加一个或若干个线程, 后续阶段按照它当前阶段所占资源比率 p 值同步增加, 即增加 $x*pi(1<i<=n)$ (x 表示第一阶段增加数).

2　基于SEDA的企业服务总线的设计与实现

2.1 企业服务总线阶段划分

把上述 SEDA 阶段架构运用到企业服务总线的设计中. 根据企业服务总线一般功能模块划分, 可分五个阶段: 消息监听、协议解析、消息转换、消息路由、消息发送[5,6]. 企业服务总线框架如图 2 所示.

考虑到版面的布局, 图 2 中省略了协议解析、消息转换、消息路由三个阶段.

从图 2 中可以看出, 申请服务的请求信息被封装成事件, 进入消息监听模块, 然后经过协议解析、消息转换、消息路由、消息发送处理几个阶段, 每一阶段都有自己的事件处理器和线程池, 阶段控制器负责调整本阶段线程池的大小.

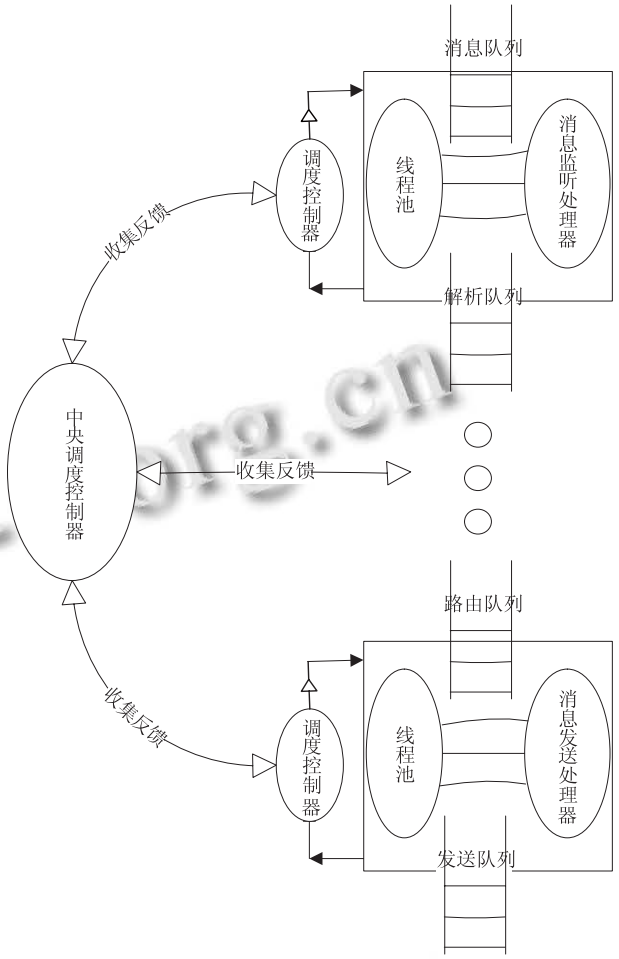

图 2　基于 SEDA 的企业服务总线框架图

2.2 企业服务总线中央调度器设计

中央调度器是一个守护线程, 在系统启动时, 将会被调用. 中央调度器定时采集(比如时间间隔为 5 秒)各个阶段的负载情况信息, 经过计算负载均值和方差判断系统的阶段负载情况.

中央调度器根据算法, 得出各阶段事件队列和线程池大小的调整参数, 再把参数信息分发到企业服务总线五个阶段的调度控制器中, 由阶段调度控制器根据收到的参数进行调整.

3　实验结果及分析

本文对上述负载控制设计的企业服务总线系统和基于常规线程池设计的企业服务总线系统进行了相关性能测试.

ESB 服务器配置如下: 2.66GHz CPU 2G 内存, CentOs5.5 操作系统, 在百兆局域网内进行测试. 此次

测试通过并发不同数量的客户端服务请求情况下, 对 ESB 平均响应时间和阶段线程大小变化进行观察. 基于常规线程池设计的企业服务总线的线程池大小固定在 30 个, 基于负载控制的 SEDA 企业服务总线各个阶段线程池的初始值为 30 个. 测试方法在通过并发不同数量的客户端请求服务的情况下, 对 ESB 平均响应时间和各个阶段线程池大小进行比较. 实验测试数据如图 3, 表 3 所示.

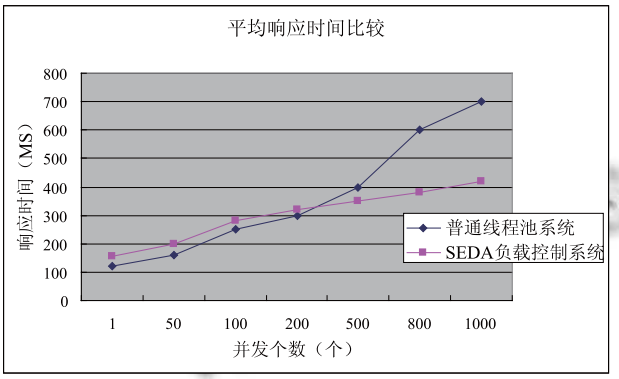

图 3 平均响应时间比较

从图 3 可以看出, 在客户端申请服务并发数较低的情况下, 基于负载控制的系统性能并不优于常规线程设计的系统. 这主要是由于基于负载控制的系统, 需要额外维护一个守护线程来收集各个阶段的运行状况, 这个过程中涉及到访问各个阶段的公共变量, 需要加锁和解锁的操作. 但随着长时间的运行, 它要比基于常规线程池设计的系统更加稳定.

表 3 系统负载状态表

时刻\线程池大小	消息监听	协议解析	消息转换	消息路由	消息发送
T1	30	30	30	30	30
T2	30	28	30	31	30
T3	31	25	27	32	29
T4	32	20	22	33	28

由表 3, 可以看出协议解析和消息转换两个阶段的事件处理相对比较节省资源. 因为根据系统代码的结构, 这两个阶段只需在内存中进行相应计算即可. 而消息监听和消息路由两个事件处理起来比较耗时, 主要是由于消息监听阶段主要负责 socket 的读取, 消息路由需要查询路由表, 在根据路由算法选择路由, 它们将会占用一些时间. 由此可见, 基于负载控制来调整线程池大小的系统要比基于常规静态线程池的系统设计线程池大小上更加灵活, 系统稳定性更好.

4 结语

通过基于阶段事件驱动负载控制设计出的系统, 可以了解一个完整的任务在系统处理过程时, 哪些阶段是耗费资源阶段, 从而根据系统运行实际情况分配线程个数, 提高系统运行效率. 后续阶段的研究主要工作是进一步研究系统运行时, 各个阶段的线程池状态, 且对每个阶段的公共数据块的设计进行细化, 以减少在对全局数据操作时对加锁的依赖.

参考文献

1 王晓明,牛立栋.基于 SOA 的企业应用集成技术分析.无线电工程,2012,(1):54–57.

2 谢继晖,白晓颖,陈斌等.企业服务总线研究综述.计算机科学,2007,(11):13–18.

3 Matt Welsh, David Culler, Eric Brewer, SEDA: An architecture for highly, well-conditioned internet services. ACM SIGOPS Operating Systems Review, 2001, 35(5): 230–243.

4 罗海南.多阶段事件驱动架构性能调优机制的研究[硕士学位论文].成都:电子科技大学,2009.

5 Fernandez EB. Two patterns for distributed systems: enterprise service bus (ESB) and distributed publish/subscribe. 18th Conference on Pattern Languages of Programs, PLoP.

6 董率,廉东本,刘鹏.基于 SEDA 的企业服务总线的设计与实现.计算机系统应用,2010,19(9):44–48.